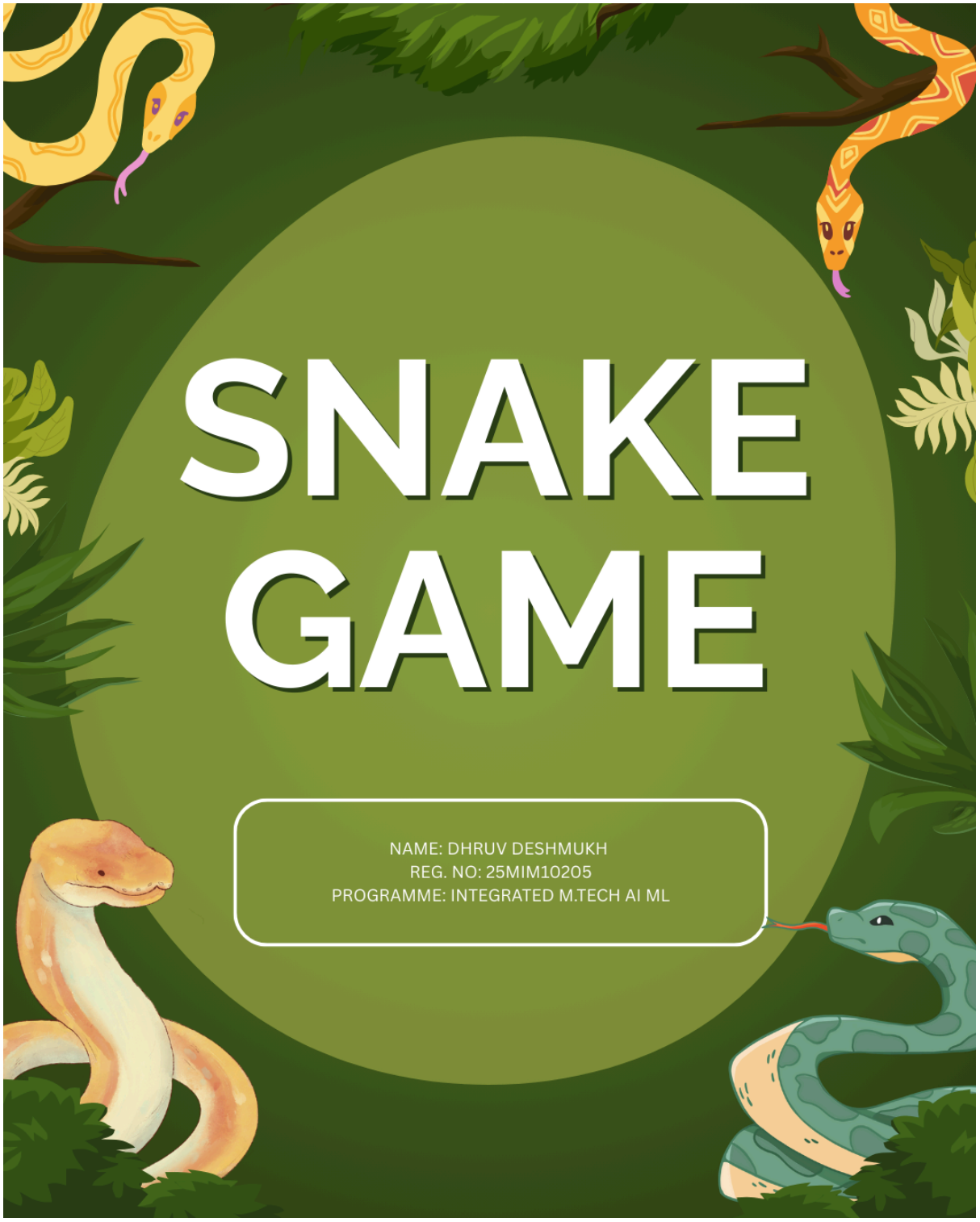
# SNAKE GAME

(VITYARTHI ASSIGNMENT)

NAME: DHRUV DESHMUKH REG NO. 25MIM10205



# Introduction:

The Snake Game is a visually enhanced version of the classic snake arcade game developed using Python’s turtle graphics. The game challenges players to control a snake that moves in zigzag patterns to collect multiple apples while avoiding collisions with walls and its own body. Features like animated eyes, a flickering tongue, gradient backgrounds, and multiple food items make the gameplay more engaging and fun. This project demonstrates Python programming skills in real-time animation, event handling via keyboard controls, collision detection, and score management.

# Project Objective Statement:

The objective of this project is to build an interactive and user-friendly Snake Game using Python's turtle module that showcases smooth animation and responsive controls. The game aims to:

- Implement movement controls to navigate the snake in four directions with a unique zigzag motion pattern.
- Manage multiple food items for increased challenge and score opportunities.
- Track and display both current and high scores to motivate players.
- Include visual enhancements such as gradient backgrounds, snake eyes, and tongue animation to improve user experience.
- Strengthen understanding of graphical rendering, event-driven programming, collision detection, and dynamic updates using Python.

This project serves as both an entertaining game and a practical learning exercise in Python game development.

# Real-World Problem Identification:

Many people seek simple and engaging ways to practice programming logic, event handling, and graphical programming concepts. Classic games like Snake provide a familiar context to explore control flow, real-time updates, collision detection, and user interaction. Additionally, building such games helps address the challenge of creating smooth animations and responsive controls in beginner-friendly programming environments. This project identifies the need for practical, hands-on coding exercises that bring together multiple programming fundamentals into a cohesive, fun application.

# Technical Solution Design:

This project uses Python's `turtle` graphics library to develop an interactive Snake Game that incorporates several technical elements to solve the problem of creating an engaging and intuitive user experience:

- Graphical Interface: The game window with gradient backgrounds and animated snake features (eyes, flickering tongue) provides a visually appealing environment using turtle's drawing and animation capabilities.
- Event-Driven Control: Keyboard input handling for snake movement allows dynamic user interaction and control updates.
- Game Logic: The snake moves in a zigzag pattern with a toggle offset for smooth motion. Food items (apples) are scattered randomly on the screen, and consuming them increases the snake length, speed, and score.
- Collision Detection: Checks for boundary hits and self-collisions reset the game state appropriately, ensuring challenge and fairness.
- Data Management: Score and high-score tracking are implemented to maintain player progress and motivation.
- Modular Design: Functions organize the code for movement, updating features (eyes, tongue), collision detection, and score updating to enable easy understanding and potential extensibility.

Together, these design choices form an effective technical solution that addresses both the programming learning goals and the user's desire for an enjoyable gameplay experience.

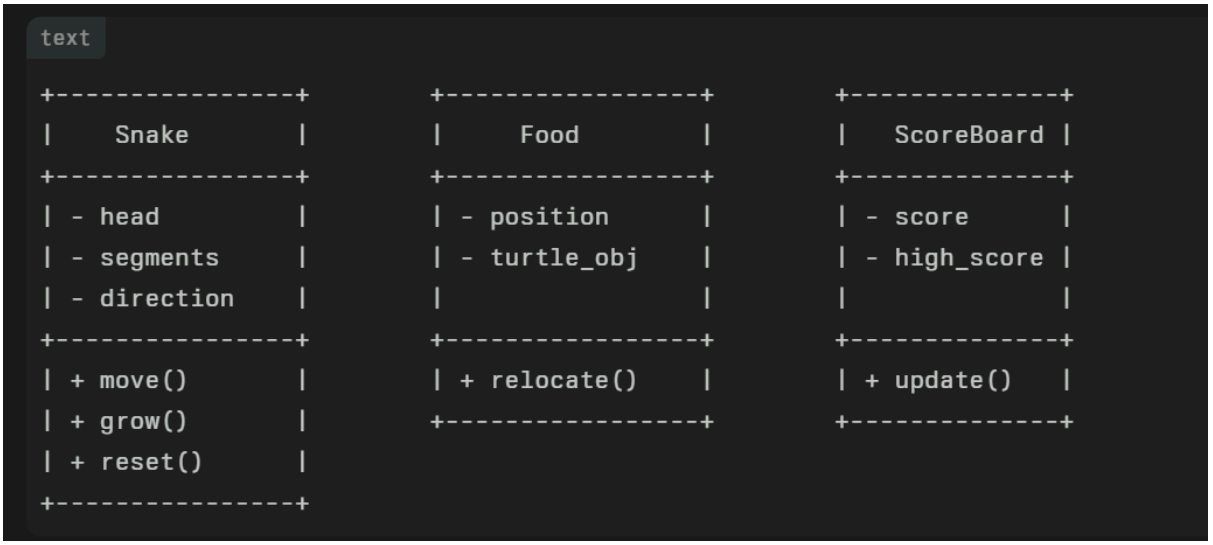
# UML diagrams based on SNAKE GAME:

## Use Case Diagram

● Actors: Player

● Use Cases: Move Snake, Eat Food, Grow Snake, Update Score, Reset Game, View Scores

### Class Diagram (Simplified)




### Sequence Diagram (Simplified)

Player → Game Loop: Input direction (W/A/S/D)
Game Loop → Snake: move()
Game Loop → Food: check_collision()
Game Loop → Snake: grow()
if collision Game Loop → ScoreBoard: update()
Game Loop → Snake: check_collision() (self/border)
Game Loop → Snake: reset() if collision

## Functional Requirements:

- The game must allow the player to control the snake's movement using keyboard inputs (W, A, S, D) to move up, left, down, and right respectively.
- The snake should move continuously in the chosen direction with a zigzag motion.
- The game must randomly place three food items (apples) on the screen for the snake to eat.
- When the snake eats food, it should grow by adding a new segment to its body.
- The score should increase by 10 points for every food eaten.
- The game should keep track of the current score and the highest score achieved during the session and display them on the screen.
- The game must detect collisions with the border or the snake’s own body and reset the game accordingly.

- Visual elements like snake eyes, tongue animation, and gradient background must update dynamically during gameplay.

## Non-Functional Requirements:

- The game should run smoothly with minimal delay between frames (initial delay around 0.1 seconds).
- Graphics and animations should be visually appealing and clear.
- Controls must be responsive and intuitive.
- The code must handle resource management properly (e.g., hiding unused turtle objects off-screen).
- The game should be robust against errors such as boundary overflow and self-collision by resetting gracefully.
- The design should support easy maintenance and scalability through modular functions.
- The game should work consistently on standard Python installations with the turtle library.

## Testing Approach:

- Functional testing by running the game to verify movement controls (W, A, S, D) respond correctly without moving in the opposite direction immediately.
- Collision testing to ensure hitting the window border or the snake's body resets the game properly.
- Verification of food collision detection by checking that the snake grows and the score updates when eating any of the three food items.
- Visual inspection of dynamic elements such as the snake's eyes, flickering tongue, gradient background, and zigzag motion to ensure smooth animation.
- Testing score and high-score updating and display correctness after eating food and on game resets.
- Stress testing by playing continuously to observe no crashes, memory leaks, or object clutter from growing segments.
- Cross-platform testing to ensure compatibility with standard Python and turtle library environments.

## Challenges Faced:

- Implementing smooth zigzag movement while keeping the snake's direction logically consistent.
- Managing multiple food objects and their random repositioning without overlapping or causing erratic behavior.
- Coordinating animated features like eyes and flickering tongue to update correctly with snake movement.

- Handling collision detection precisely to avoid false triggers or missed collisions, especially during quick turns.
- Maintaining game performance and preventing lag or freezing during rapid movement or long gameplay sessions.
- Designing clear and readable code with modular functions despite the complexity of multiple moving parts and animations.

## Learnings & Key Takings:

- Gained experience with Python's turtle graphics library including custom shapes, color modes, and animation techniques.
- Developed event-driven programming skills using keyboard listeners and real-time screen updates.
- Enhanced understanding of collision detection algorithms and game state management.
- Learned how to organize code modularly for readability, maintenance, and future feature enhancements.
- Experienced blending creative visual effects with core gameplay mechanics for a more engaging user experience.
- Realized the importance of balancing game speed, responsiveness, and visual smoothness for player satisfaction.
- Improved debugging and testing skills, especially for graphical and interactive applications.

## Future Enhancements:

- Implement smoother and more varied snake movement patterns beyond simple zigzagging for added gameplay complexity.
- Add sound effects for eating food, collisions, and game over to enhance player immersion.
- Introduce different difficulty levels with varying speeds, obstacles, or power-ups.
- Create a pause and resume functionality for better user control over the game session.
- Add a menu interface with options to restart, change settings, or view instructions.
- Implement saving and loading of high scores persistently across game sessions.
- Expand the game with multiplayer support or AI-controlled snake opponents.
- Use object-oriented programming principles to better organize game objects and logic for easier scalability.

- Include more diverse and animated food items, each with different point values or effects.
- Optimize collision and movement logic for performance improvements during long sessions.
- Add touch control or swipe support for mobile or touchscreen compatibility.

# References:


- Tutorials and guides on creating snake games with Python turtle from GeeksforGeeks and DataCamp
- Various online resources and community examples for real-time game development with Python turtle
- Code snippets and programming concepts inspired by open-source educational projects and tutorials on interactive graphics and user input handling using Python turtle libraries